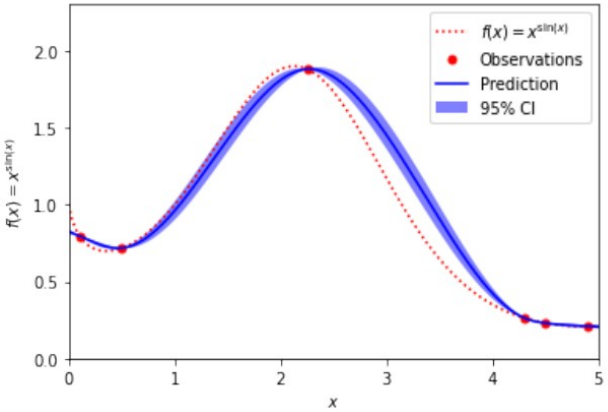
RECAP: SURROGATES & GP

FOR STRUCTURAL ESTIMATION AND UNCERTAINTY QUANTIFICATION

University of Geneva
March 25th, 2025

https://github.com/sischei/Deep_Learning_Geneva_2025

Simon Scheidegger
simon.scheidegger@unil.ch



I. Confronting Models to data

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3885021

- Contemporary models very rich (many endogenous states, exogenous states, strong non-linearities, lots of parameters,...).

- Expensive to compute.

- Consequently, economists are often forced to sacrifice certain features of the model in order to reduce model dimensionality.
  - estimate only a partial set of parameters while prefixing the others.
  - estimate the model only once using the full sample.

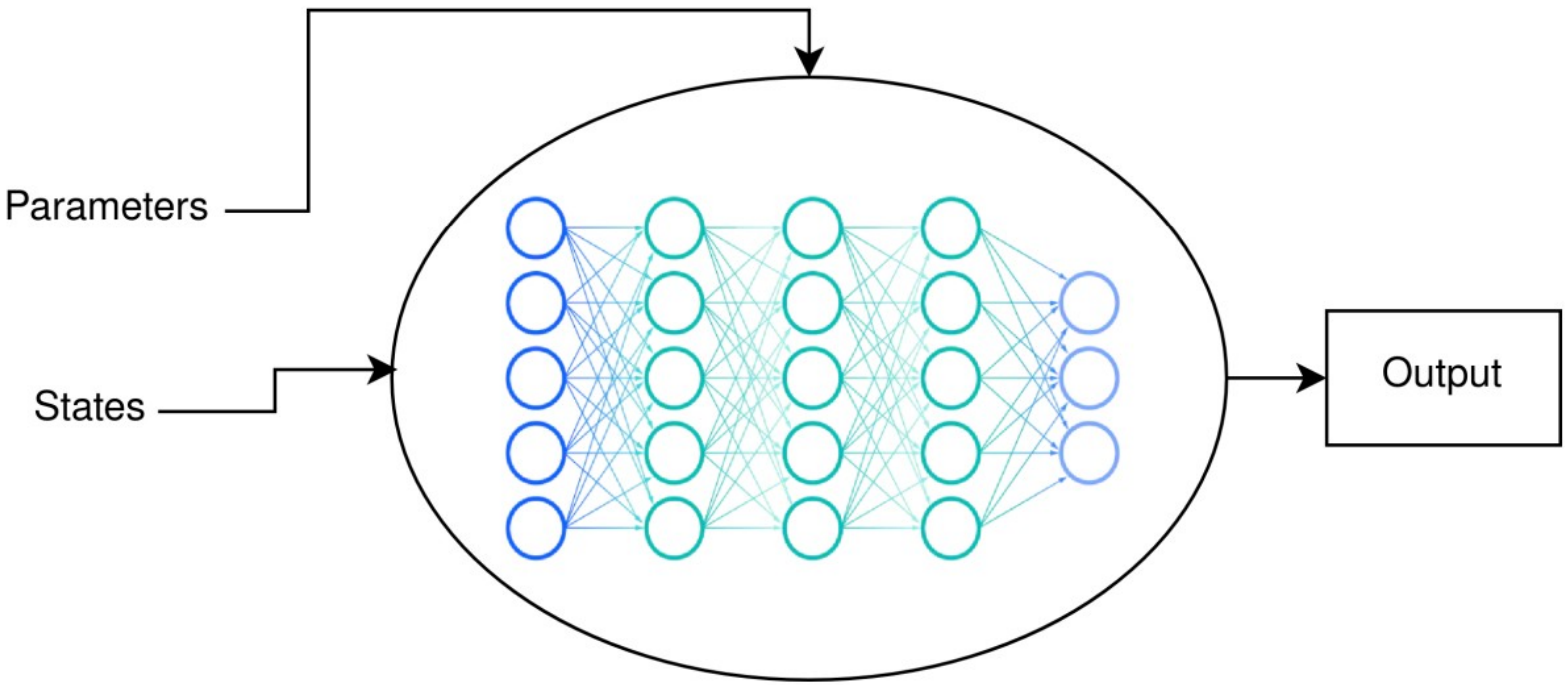
- The high computational costs limit a researcher's ability to carry out a variety of important model analyses.

- Model **estimation**, **calibration**, and **uncertainty quantification** can be daunting numerical tasks

  → because of the need to perform sometimes hundreds of thousands of model evaluations to obtain converging estimates of the relevant parameters and converging statistics

  (see, e.g., Fernández-Villaverde, Rubio-Ramrez, and Schorfheide, 2016; Fernandez-Villaverde and Guerrn-Quintana, 2020; Iskhakov, Rust, and Schjerning, 2020; Igami, 2020, among others).

Surrogate Models: 21st Century "Lookup Table"



$$\phi(\mathbf{x}_t|\theta_{NN}) = y_t$$

The basic idea

- Replace the economic model with a surrogate!
- Consider a model $$f : \mathbb{R}^m \rightarrow \mathbb{R}^k = f(\Omega_t, H_t|\Theta) = y_t$$
  - where Ω_t is a vector of dimension ω containing the observable states
  - H_t is a vector of dimension h comprising the hidden states
  - Θ is a vector of dimension θ containing model parameters
  - y_t is a vector of dimension k comprising the predicted quantities of interest (such as simulated moments, social cost of carbon in a given year, etc.)

The basic idea (II)

- The problem is that f (·) can be computationally costly, so we wish to construct a cheap to evaluate surrogate, i.e., a Neural Network that replaces the “true” function f (·):

$$\hat{f}(\Omega_t, H_t, \Theta) = \hat{f}(X_t) = y_t$$

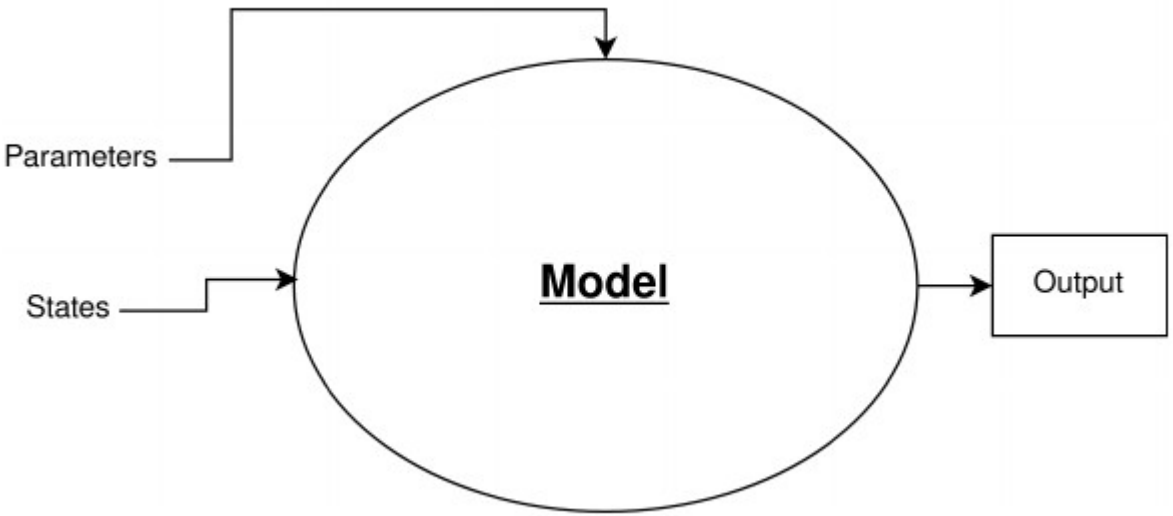
- We introduce **parameters as pseudo-state variables** (cf. Norets (2012), Scheidegger & Bilionis (2019))

  $X_t = [\Omega_t , H_t, \Theta]^T$.

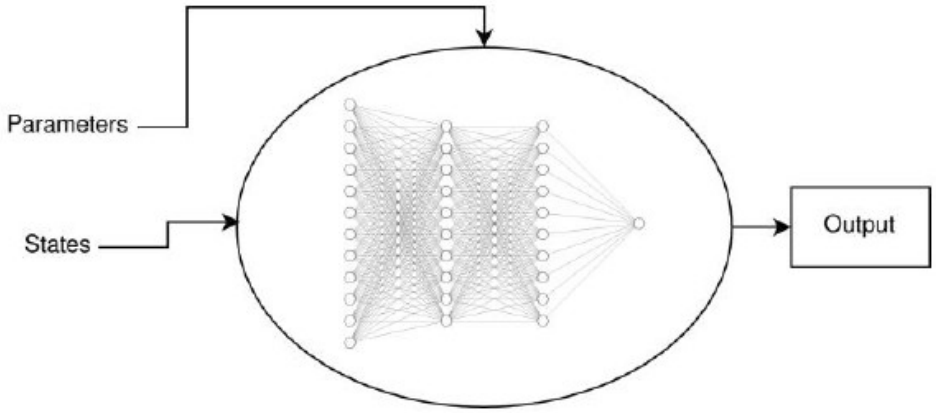
- **Solve model only once**, as a function of X_t (global solution) e.g. by using Deep Learning, e.g., by DEQN.

- For reasonable parameter ranges, you may have to use “expert knowledge”.

Why (deep) surrogate?



Why (deep) surrogate?



$$f(\text{states}, \underbrace{\text{parameters}}_{\text{Pseudo-states}}) = \text{output}$$

Some remarks

Deep surrogate is different from standard ML:

- Compared to other methods, deep neural networks are more hungry for data.
- The cost of producing a large training sample should be an important consideration.
- Unlike in standard ML, we know the true model $\Rightarrow$ unlimited data (only limited by
- computational resources); essentially no errors.
- Double descent: Use a large number of epochs
  $\rightarrow$ Stephenson and Lee (2021); Nakkiran et al., (2021)

Once trained, the deep surrogate

- is highly accurate;
- is cheaper to use by orders of magnitude; makes the gradients readily available;
- is easy to store (for 10^6 parameters - 20 MB vs. $\sim 10^6$GB when using Cartesian grid).

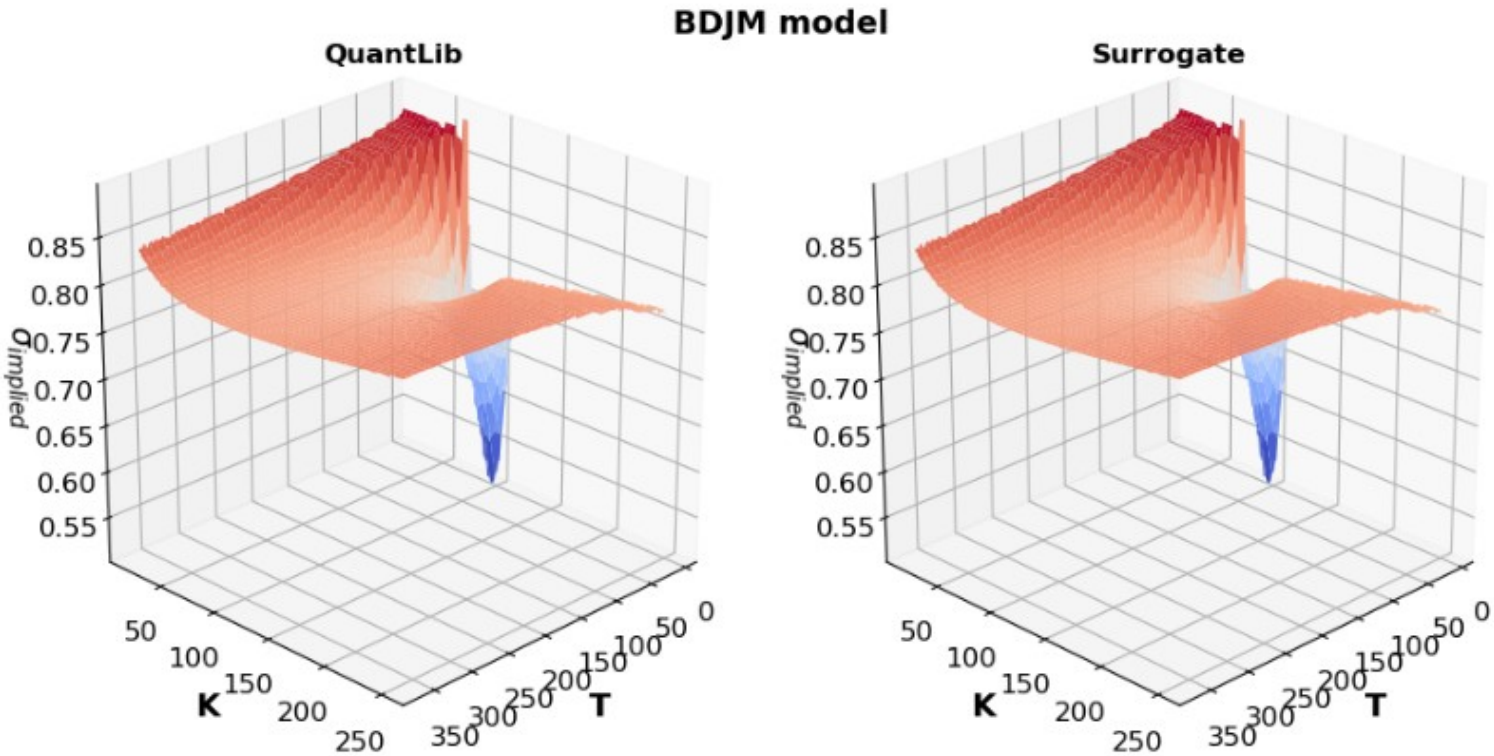
Pay the cost upfront; use it for free later.

- High quality surrogates can be shared with and build on by a community.
- Deep surrogates for workhorse quantitative economic and financial models.

Static data

https://github.com/DeepSurrogate/OptionPricing

Deep Surrogate for Asset Pricing



The figure above the implied volatility surface of the Bates model with Double Jump Exponential generated with QuantLib (left) and the Deep-Surrogate (right)

Example: DEQN with pseudo-states

- Let’s have a look at a stochastic growth-model with parameters as pseudo-states.
- Code: day2/code/DEQN_production_code/stochastic_growth_pseudostates

  → Solutions can be used to generate to simulate moments, and other quantities of interest.

II. DEQN surrogate

The planner's problem is

$$\max_{C_t, K_{t+1}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t b_t \frac{C_t^{1-\tau} - 1}{1-\tau}, \tag{1}$$

subject to the resource constraint

$$C_t = A_t K_t^{\alpha} + (1-\delta) K_t - K_{t+1} \ \left(\text{multiplier } \beta^t b_t \mu_t\right) \tag{2}$$

and the irreversability condition

$$K_{t+1} - (1-\delta) K_t \geq 0 \ \left(\text{multiplier } \beta^t b_t \lambda_t\right). \tag{3}$$

The Lagrangian takes the form

$$\max_{C_t, K_{t+1}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t b_t \left\{ \frac{C_t^{1-\tau} - 1}{1-\tau} + \mu_t \left[A_t K_t^{\alpha} + (1-\delta) K_t - K_{t+1} - C_t \right] + \lambda_t \left(K_{t+1} - (1-\delta) K_t \right) \right\}. \tag{4}$$

The Kuhn-Tucker conditions take the form:

$$(C_t) \ : \ 0 = C_t^{-\tau} - \mu_t \tag{5}$$

$$(K_{t+1}) \ : \ 0 = -\mu_t + \lambda_t + \beta \mathbb{E}_t \left\{ \frac{b_{t+1}}{b_t} \left(\mu_{t+1} \left[\alpha A_{t+1} K_{t+1}^{\alpha-1} + (1-\delta) \right] - \lambda_{t+1} (1-\delta) \right) \right\} \tag{6}$$

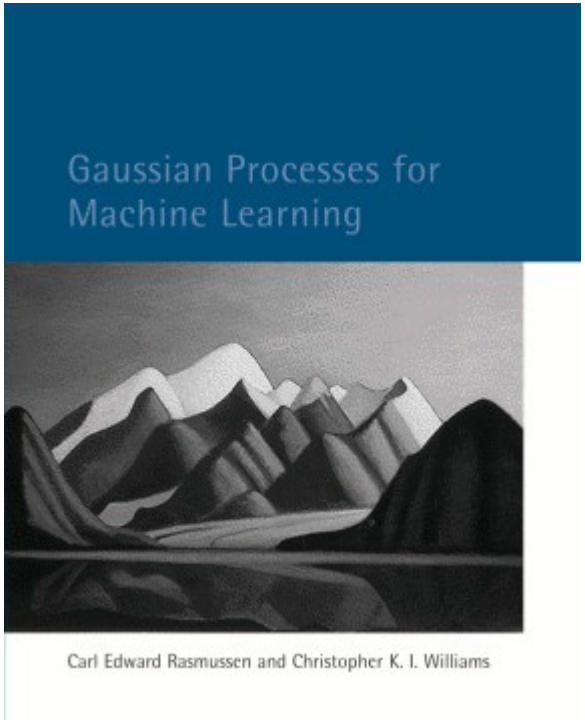
$$(CS) \ : \ 0 = \lambda_t \left[K_{t+1} - (1-\delta) K_t \right]. \tag{7}$$

Define $d_t = b_t / b_{t-1}$. We assume that the exogenous shock processes evolve according to

$$\ln A_t = (1-\rho_A) \ln A_* + \rho_a \ln A_{t-1} + \sigma_a \epsilon_{a,t} \tag{8}$$

$$\ln d_t = \rho_d \ln d_{t-1} + \sigma_d \epsilon_{d,t}. \tag{9}$$

Gaussian Process Regression



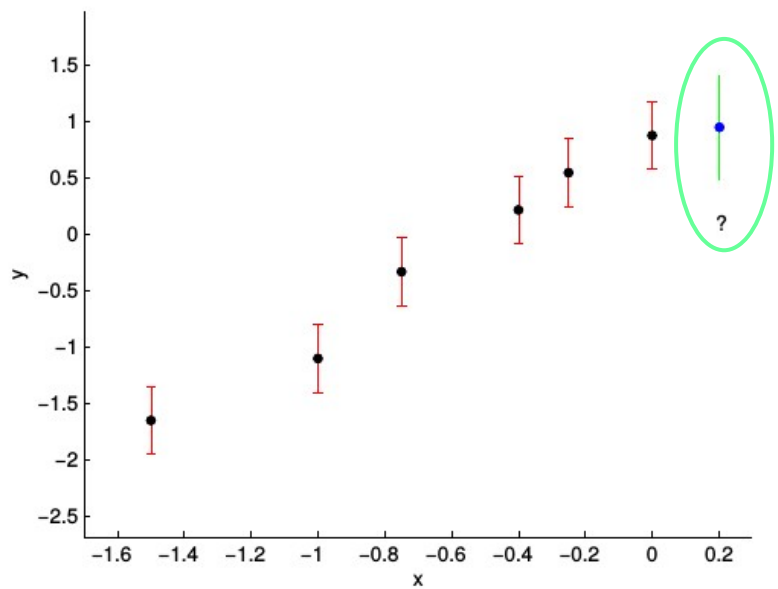
http://www.gaussianprocess.org/gpml/

Recall: Aim of Regression

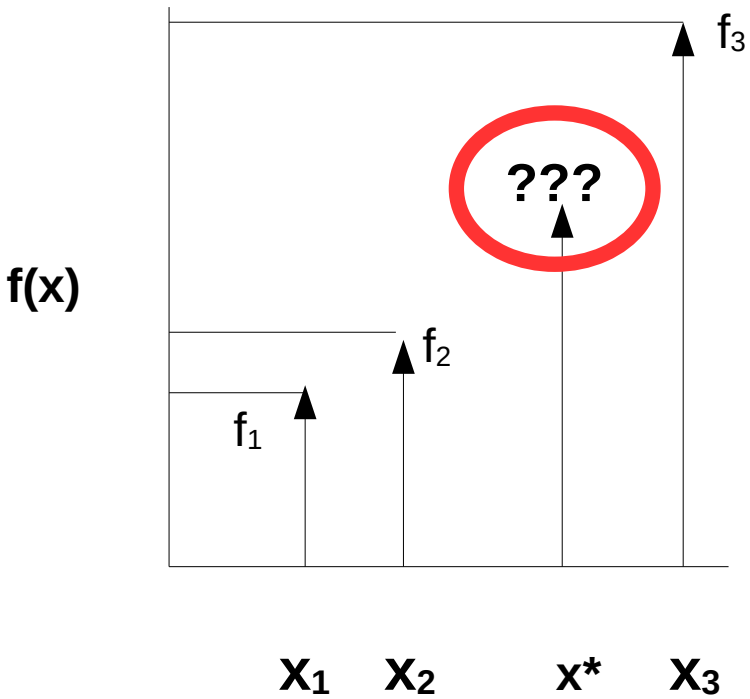
- Given some (potential) noisy observations of a dependent variable at certain values of the independent variable x, what is our best estimate of the dependent variable y at a new value, x_*?
- Let *f* denote an (unknown) function which maps inputs x to outputs

$$f: X \rightarrow Y$$

- Modeling a function *f* means mathematically representing the relation between inputs and outputs.
- Often times, the shape of the underlying function might be unknown, the function can be hard to evaluate, or other requirements might complicate the process of information acquisition.



Observations → Interpolation



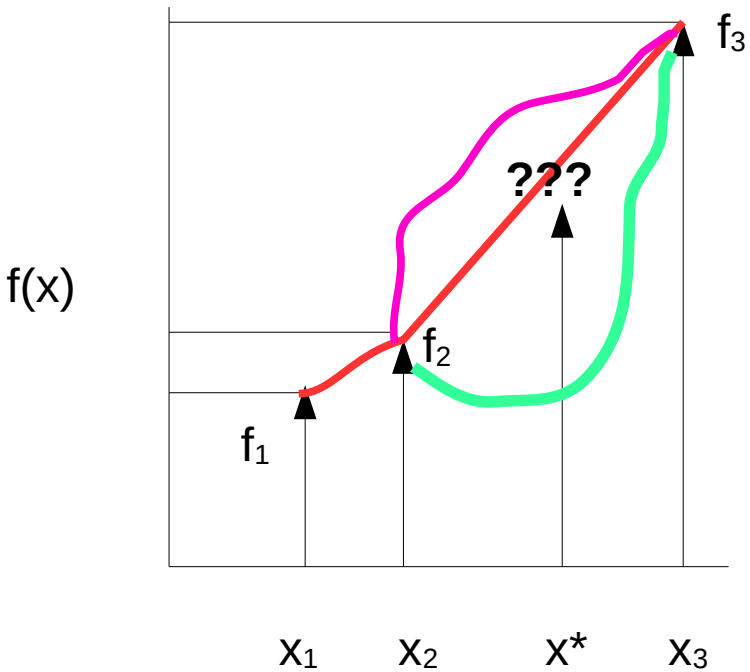
We have 3 observations at $\mathbf{x}_i$ for $\mathbf{f(x_i)}$

→ **Given the data pairs**
$D = \{ (x_1,f_1), (x_2,f_2), (x_3,f_3)\}$

→ **want to find/learn the function that describes the data, i.e., for a "new" x*, we want to know what f(x*) would be!**

Observations → Interpolation (II)

We assume that **f's (the height) are Gaussian distributed**, with **zero – mean** and some **covariance matrix K**.



$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{11}, K_{12}, K_{13} \\ K_{21}, K_{22}, K_{23} \\ K_{31}, K_{32}, K_{33} \end{bmatrix} \right)$$

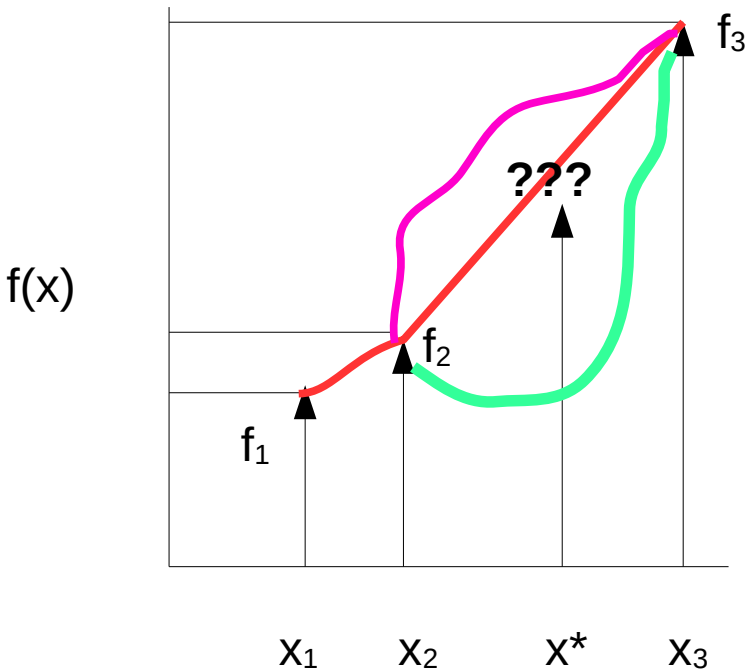
Note: f_1 and f_2 should probably be more correlated, as they are nearby (compared to f_1 and f_3).

→ The prior mean function μ reflects the expected function value at input x: $\mu(x) = \mathbb{E}(f(x))$

→ It is often set to 0.

Observations → Interpolation (II)

We assume that **f's (the height) are Gaussian distributed**, with **zero – mean** and some **covariance matrix K**.



$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{11}, K_{12}, K_{13} \\ K_{21}, K_{22}, K_{23} \\ K_{31}, K_{32}, K_{33} \end{bmatrix} \right)$$

Note: f_1 and f_2 should probably be more correlated, as they are nearby (compared to f_1 and f_3), e.g.,

$$\sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1, 0.7, 0.2 \\ 0.7, 1, 0.6 \\ 0.2, 0.6, 1 \end{bmatrix} \right)$$

Covariance matrix **constructed** by some "**measure of similarity**", i.e., a **kernel function** (parametric ansatz), such as "squared exponential". Parameters can be obtained e.g. via MLE (later).

$$\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2)$$

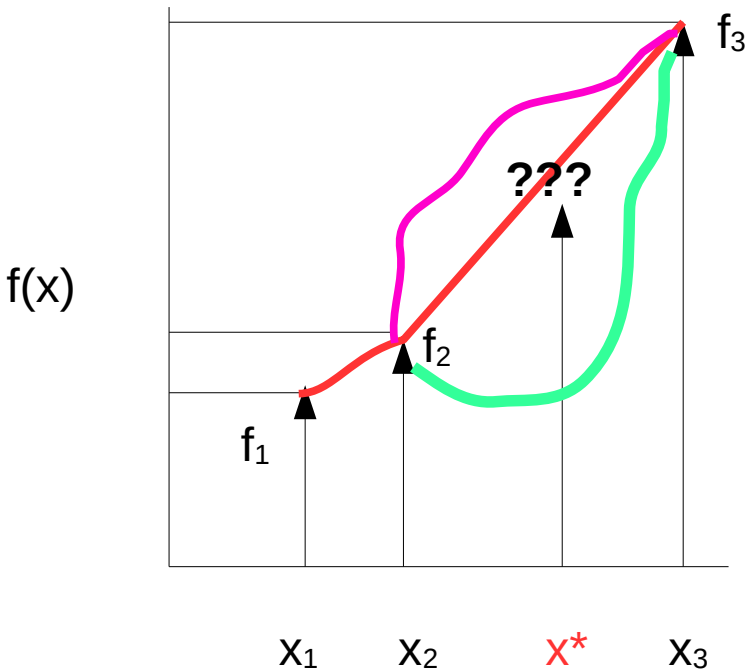
σ_f^2 – controls vertical variation.

ℓ – controls horizontal length scale.

Observations → Interpolation (III)

Given data D = { (x_1,f_1), (x_2,f_2), (x_3,f_3)} → **$f(x^*) = f_*$** ?
→ Assume f ~ N(0 , K(·,·))
→ Assume f(x*) ~ N(0, K(x*,x*))

3d-Covariance K from the training data



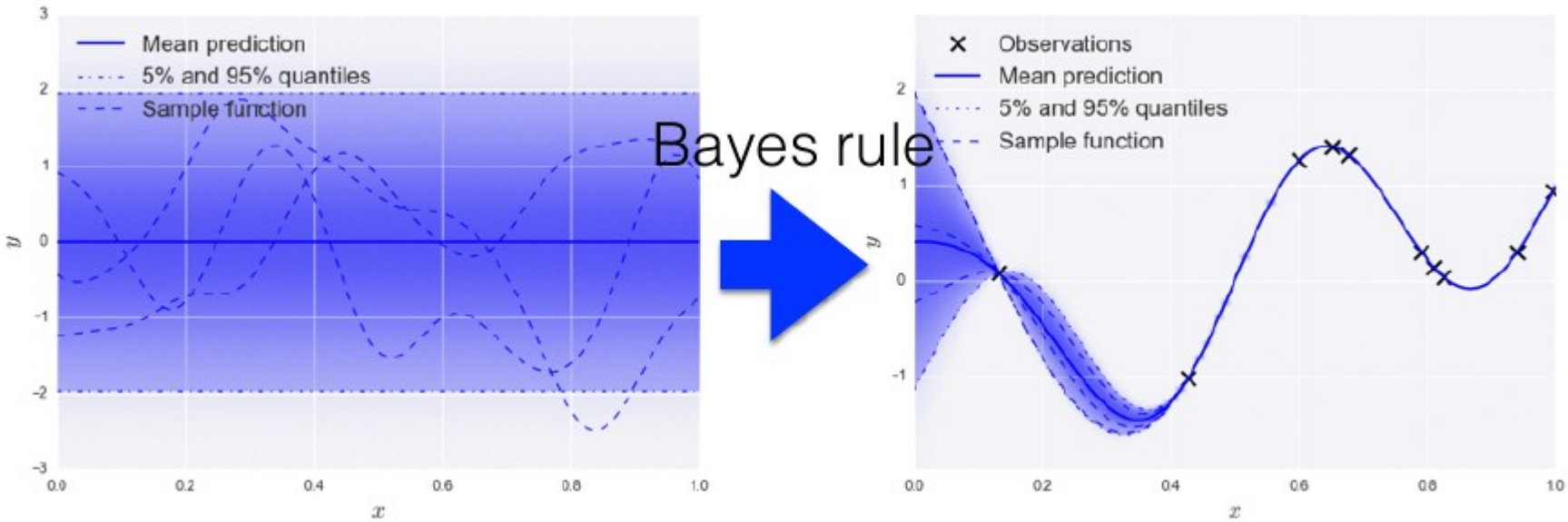
$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_* \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{11}, K_{12}, K_{13}, K_{1*} \\ K_{21}, K_{22}, K_{23}, K_{2*} \\ K_{31}, K_{32}, K_{33}, K_{3*} \\ K_{*1}, K_{*2}, K_{*3}, K_{**} \end{bmatrix} \right)$$

→ **Joint distribution over f and f_*** .
→ **We need the conditional of f_* given f**.
→ In this example, we "cut" in 3 dimensions.
→ What is left is a 1-dimensional Gaussian, i.e., the Gaussian for f_*

$$K(x_1, x_*) = K_{1*}$$

Interpolation → Noiseless GPR

(see, e.g., Rasmussen & Williams (2006), with references therein)



Prior GP

Posterior GP

Training set: $D = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \right)$$

$$\begin{aligned} p(\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f}) &= \mathcal{N}(\mathbf{f}_* | \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \\ \boldsymbol{\mu}_* &= \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1} (\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned}$$

Test point = interpolation at X*

→ **predictive mean** $\mu_* = \mathbb{E}(f_*)$

→ **Confidence Intervals!**

Where we have data, we have high confidence in our predictions.

Where we do not have data, we cannot be to confident about our predictions.

GPR with noisy data (II)

- In this case (presence of noise), the model is not required to interpolate the data, but it must come "close" to the observed data.

- The covariance of the observed noisy responses is

$$\mathrm{cov}\left[y_p, y_q\right] = \kappa(\mathbf{x}_p, \mathbf{x}_q) + \sigma_y^2 \delta_{pq}$$

where $\delta_{pq} = \mathbb{I}(p = q)$

- The second matrix is diagonal because we assumed the noise terms were independently added to each observation.

The GPR with noisy data (III)

- The joint density of the observed data and the **latent, noise-free function** on the test points is given by

Noise in the diagonal, rest is as before.

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right)$$

Latent function.

- where we are assuming the mean is zero, for notational simplicity.

- Hence the posterior predictive density is

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) &= \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \\ \boldsymbol{\mu}_* &= \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{K}_* \end{aligned}$$

Prediction at a single test point

In the case of a single test input, this simplifies as follows

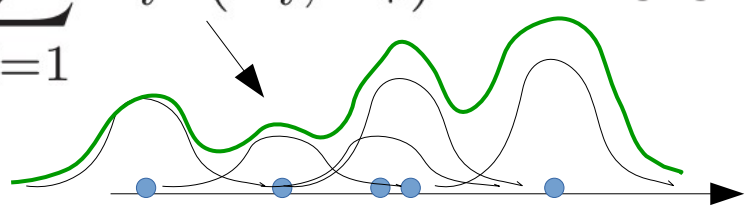
$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \quad = \quad \mathcal{N}(f_*|\mathbf{k}_*^T \mathbf{K}_y^{-1}\mathbf{y}, k_{**} - \mathbf{k}_*^T \mathbf{K}_y^{-1}\mathbf{k}_*)$$

where $\mathbf{k}_* \;=\; [\kappa(\mathbf{x}_*, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}_*, \mathbf{x}_N)]$

and where $k_{**} \;=\; \kappa(\mathbf{x}_*, \mathbf{x}_*)$ (=1)

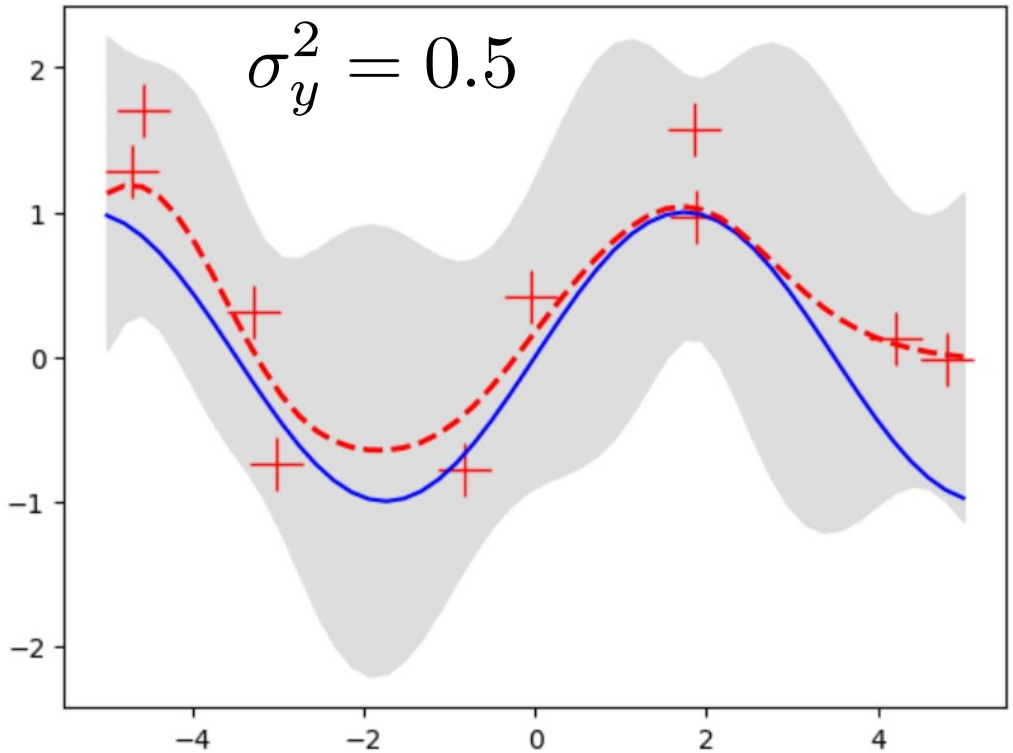
Again, we can write the posterior mean as expansion of basis functions

$$\overline{f}_* = \underbrace{\mathbf{k}_*^T}_{1 \times N} \underbrace{\mathbf{K}_y^{-1}}_{N \times N} \underbrace{\mathbf{y}}_{N \times 1} = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_*) \quad \text{where } \boldsymbol{\alpha} = \underbrace{\mathbf{K}_y^{-1}\mathbf{y}}_{\text{from training data}}$$



Some Plots

cf. demo/1d_gp_example.ipynb



- Even in the regions where you have data, there is still uncertainty.
- In the noise-free version of GPR, the uncertainty is 0 at observation points.
- But we still have the same properties as before: where we have data, we are more certain compared to the case where we have no data.

Noise improves numerical stability

- It is common to use small noise even if there is not any in the data.
- Cholesky fails when covariance is close to being semi-positive definite.
- Adding a small noise improves numerical stability.
- It is known as the “jitter” or as the “nugget” in this case.

“Learning” the kernel parameters

- To estimate the kernel parameters, we could use exhaustive search over a discrete grid of values, with validation loss as an objective, but this can be quite slow.
- Here we consider an empirical Bayes approach, which will allow us to use continuous optimization methods, which are much faster.
- In particular, we will maximize the marginal likelihood.

Example

- lectures/day4/code/01_recap_week1.ipynb